

Beijing institute of technology
School of computer science

DATA MINING

**RELATIONAL RETRIEVAL USING
RANDOM WALK WITH
RESTART(RWR)**

MARWAN ANISS QUDAR **2820150124**

ELFATHI ABDALLAH HASSAN **2820150087**

JOHN BETER **2820150014**

Relational Retrieval Using Random Walk With Restart(RWR)

Introduction:

Accessing the scientific literature in the past researches has focused on a small number of well-defined tasks which represent the scientific literature as a set of documents such as named entity recognition(NER) and normalization . In fact, scientific literature naturally includes substantial metadata such as author names, citations, and publication venues, as well as derived metadata (such as gene and protein names, in the biomedical literature).

An alternative way to represent the scientific literature is as a labeled directed graph, with typed nodes representing documents, terms, and metadata, and labeled edges representing the relationships between them (e.g., “authorOF”, “datePublished”,etc).

Representing the scientific literature as a labeled directed graph is an effective way , which the typed nodes are representing documents, terms, and metadata , and labeled edges representing the relationships between them (e.g. “authorOF”, “datePublish”,etc). And by implementing the labeled graph we will be able to analyze the entities in any specific dataset type and build the network between each entity according to the connection between them .

By building the network , the general structure will be obvious and that leads to retrieval the specific knowledge that tasks can be implemented on it to improve and get different results from it by applying specific methods for reaching the required results and achieving the main goal .

Dataset:

The data in our project, is a dataset which is in the format of “.fly “ this dataset a biological literature graph which is an integrated database for Drosophila and Anopheles genomics, and contains about 127K papers tagged with genes and proteins. In fact, scientific literature for example “.fly” naturally includes substantial metadata such as author names, citations, and publication venues, as well as derived metadata (such as gene and protein names, in the biomedical literature). The way to represent the scientific literature is as a labeled directed graph, with typed nodes representing documents, terms, and metadata, and labeled edges representing the relationships between them (e.g., “authorOf”, “datePublished”, etc).

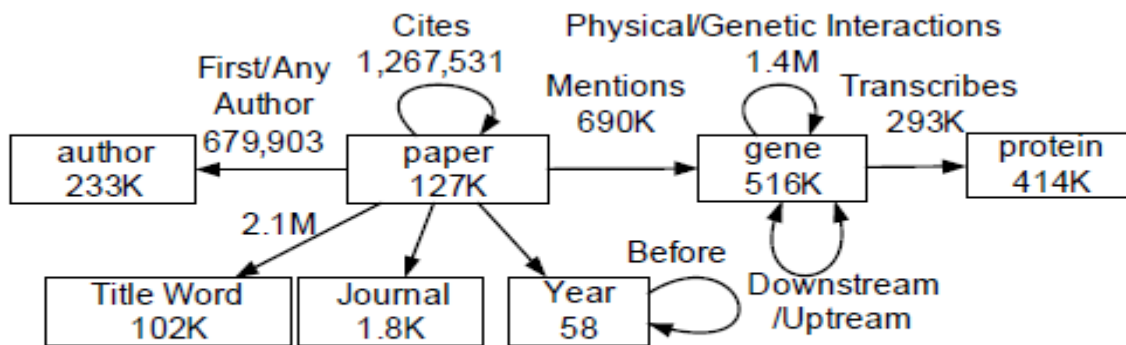


Fig. 2. Schema of the fly data.

Structure of the Fly dataset :

Year(18791224,2008)

Journal(18791224,Genetics)

Author(18791224,Vandre_CL Kamakaka_RT Rivier_DH)

AuthorF(18791224,Vandre_CL)

AuthorL(18791224,Rivier_DH)

Title(18791224,dna end-binding protein ku regulates silencing internal hml hmr loci saccharomyces cerevisiae)

Pre-processing :

After opening the dataset and analyzing its components , we found different relationships between the entities like a paper has a specific year and this paper has been published by different author in a specific journal. So as we can see there are different connections between each component in the dataset.

And then we came with a new task to implement in the dataset to get new results that will help to understand new knowledge . If we come back to the structure of the dataset we will find that the each author has co-author that they worked together to publish different papers and each author has worked with different authors , so it is kind of impossible to know who is working with other according to the massive information that the dataset contains .

so what we have done is to specify each author and see who is working with that author we specified them as Main Author and Co-Author . And by implementing this task it will be easier to know what is the relationship with each author and who is working in each other in publishing the scientific papers.

1- Specifying the main authors :

In the beginning we tried to get the list of all authors in the dataset because it was very hard to get them from the main structure , so we implemented this code :

```
import sys

data_path = r'C:\Users\Maro_Qudar\Desktop\new_ML\alchemy\AbstractInfor.db'
author_path = r'C:\Users\Maro_Qudar\Desktop\new_ML\alchemy\AuthorCopy.txt'
key_word = 'Author'
notkey1 = 'LAuthor'
notkey2 = 'FAuthor'
dataFP = open(data_path, 'r')
authorFP = open(author_path, 'w')
for eachline in dataFP:
    #print eachline
    if key_word in eachline and notkey2 not in eachline and notkey1 not in e
        authorFP.write(eachline)
dataFP.close()
authorFP.close()
```

And then we generating the list in a new file and here is a sample of the authors that has been generated :

Author(18791224,Vandre_CL Kamakaka_RT Rivier_DH)
Author(17434796,Schulz_TA Prinz_WA)
Author(7958899,Ozer_J Moore_PA Bolden_AH Lee_A Rosen_CA Lieberman_PM)
Author(11327883,Waterborg_JH)
Author(1918023,Woychik_NA Lane_WS Young_RA)
Author(12381738,Lechner_E Xie_D Grava_S Pigaglio_E Planchais_S Murray_JA
Parmentier_Y Mutterer_J Dubreucq_B Shen_WH Genschik_P)
Author(2167439,Buchman_C Skroch_P Dixon_W Tullius_TD Karin_M)
Author(14523396,Pintard_L Peter_M)
Author(7037542,Ter-Avanesian_MD Shubochkina_EA Inge-Vechtomov_SG)
Author(15153069,Forsgren_M Attersand_A Lake_S Grunler_J Swiezewska_E Dallner_G
Climent_I)
Author(9719874,Scott_SV Klionsky_DJ)
Author(17411056,Pastore_A Martin_SR Politou_A Kondapalli_KC Stemmler_T Temussi_PA)

So in here for example Author (paper_id, name of the authorF and author L)

AuthorF : First Author .

AuthorL: Last Author.

Al-Aidroos_K
Holly_JA
Datson_NA
van_Nuland_NA
Cloud_KG
Hawley_RS
Kwast_L
Haynes_SR
Egashira_N
Scrimale_T
Tadauchi_T
Schofield_DA
Takiguchi_S
Pan_SM
Chan_W
Kiyono_K
Krol_AA
Robert_F
Arenas_JE

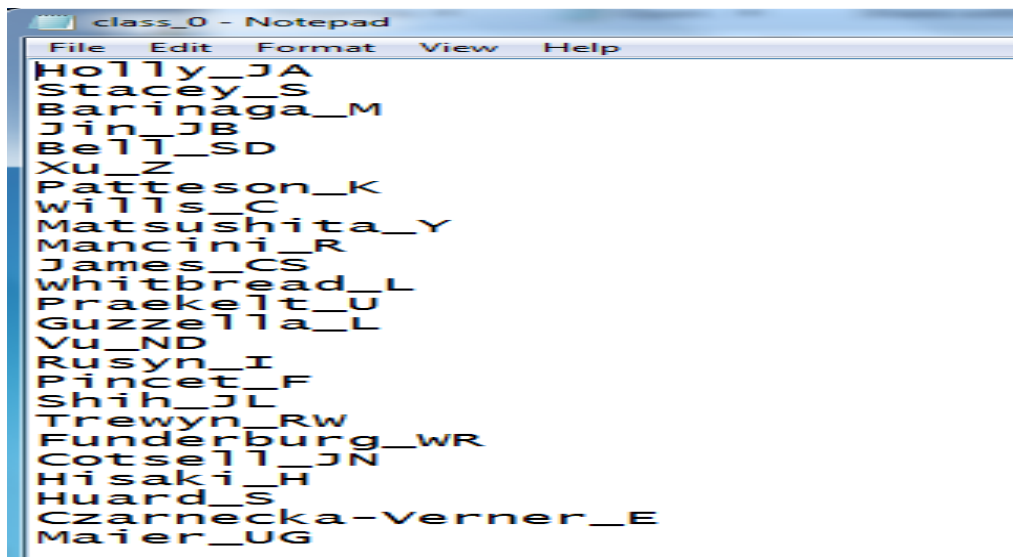
The file generated more than 69000 authors in the dataset , so as you can see the big number of the authors which is very hard to get the Author and the Co-author from analyzing the dataset line by line instead of using a specific methods and techniques to get that knowledge .

2- Making the main clusters :

So after we get the authors , we had to divide them into different clusters so is going to be easier to work with this data. We have around 70000 authors which we will need 100 cluster and each cluster will contain 700 authors

```
# get separation name between main authors and coauthors
UserTimes_dict = {}
for eachdata in follow_data:
    if eachdata.strip('\n') == "":
        continue
    two_user = eachdata.strip('\n').split(" ")
    # all_author.append(two_user[0])
    # all_author.append(two_user[1])
    if UserTimes_dict.has_key(two_user[0]):
        if UserTimes_dict[two_user[0]].has_key(two_user[1]):
            UserTimes_dict[two_user[0]][two_user[1]] = UserTimes_dict[two_user[0]][two_user[1]] + 1
        else:
            UserTimes_dict[two_user[0]].setdefault(two_user[1],1)
    else:
        UserTimes_dict.setdefault(two_user[0],{two_user[1]:1})
# for key in UserTimes_dict.keys():
#     print 'key = %s' % key,UserTimes_dict[key]
```

Here is a class number 0 as a sample :



```
class_0 - Notepad
File Edit Format View Help
Holly_JA
Stacey_S
Barinaga_M
Jin_JB
Bell_SD
Xu_Z
Patteson_K
Wills_C
Matsushita_Y
Mancini_R
James_CS
Whitbread_L
Praekelt_U
Guzzella_L
Vu_ND
Rusyn_I
Pirnet_F
Shih_JL
Trewyn_RW
Funderburg_WR
Cotsell_JN
Hisaki_H
Huard_S
Czarnecka-Verner_E
Maier_UG
```

But the problem is that these classes has been chosen randomly and that is not effective because we want to get the author and his co-authors by accurate results ,

so we have applied the K-means algorithm to build the clusters in a more accurate way . We get each author from each class and then calculate its corporation times between main author and co-author through matrix analysis for each generated class and then we get the stable results after applying K-means for many iterations .

Here is the code to calculate the corporation times between main author and co-author :

```
UserTimes_dict = {}
for eachdata in follow_data:
    if eachdata.strip('\n') == "":
        continue
    two_user = eachdata.strip('\n').split(" ")
    # all_author.append(two_user[0])
    # all_author.append(two_user[1])
    if UserTimes_dict.has_key(two_user[0]):
        if UserTimes_dict[two_user[0]].has_key(two_user[1]):
            UserTimes_dict[two_user[0]][two_user[1]] = UserTimes_dict[two_user[0]][two_user[1]] + 1
        else:
            UserTimes_dict[two_user[0]].setdefault(two_user[1],1)
    else:
        UserTimes_dict.setdefault(two_user[0],{two_user[1]:1})
```

Here is the code for applying K-means :

```

def calSimi(node, class_i):
    simi = 0
    for eachnode in class_i:
        siminode = 0
        author1 = node
        author2 = eachnode
        if UserTimes_dict.has_key(author1):
            if UserTimes_dict[author1].has_key(author2):
                siminode = UserTimes_dict[author1][author2]
            else:
                siminode = 0
        elif UserTimes_dict.has_key(author2):
            if UserTimes_dict[author2].has_key(author1):
                siminode = UserTimes_dict[author2][author1]
            else:
                siminode = 0
        simi = simi + siminode
    #return simi
    return float(simi)/float(len(class_i))

```

```

N = 7
count = 0
while count < N:
    changtimes = 0
    for key in authorClass_dict.keys():
        #print key, authorClass_dict[key]
        maxsimi = 0
        ori_class = authorClass_dict[key]
        max_class = ori_class
        class_list[ori_class].remove(key)
        for i in range(0, classNum):
            simi = calSimi(key, class_list[i])
            if simi > maxsimi:
                maxsimi = simi
                max_class = i
        if max_class != ori_class:
            changtimes = changtimes + 1
            authorClass_dict[key] = max_class
            class_list[max_class].append(key)
        #print key, authorClass_dict[key]
    print changtimes
    print "the", count, "iteration done!!!"
    count = count + 1
    if changtimes < 50:
        break

```


3- Applying Random Walk with Restart (RWR):

We have considered a random walk particle that starts from node i . The particle iteratively transmits to its neighborhood with the probability that is proportional to their edge weights. So we applied RWR in the clusters that have been generated to get the top 3 corporation authors by using Markov Chain for the matrix calculations in each cluster .

The relevance score defined by RWR has many good properties such as it can capture the global structure of the graph , it can capture the multi-facet relationship between two nodes.

```
def randomWalk(matrix_Normalized):
    conTimes_iter = matrix_Normalized
    while 1:
        last_iter = conTimes_iter
        conTimes_iter = np.dot(conTimes_iter,matrix_Normalized)
        if (conTimes_iter - last_iter).sum() < 1:
            break
    return conTimes_iter

def randomWalkReset(matrix_Normalized,author_i):
    alpha = 0.25
    M,N = matrix_Normalized.shape
    if author_i >= int(M):
        print 'over the range!'
        return 0
    matrix_Reset = matrix_Normalized*(1.0-alpha)
    matrix_Reset[:,author_i] = matrix_Reset[:,author_i] + alpha
    conTimes_iter = matrix_Reset
    while 1:
        last_iter = conTimes_iter
        conTimes_iter = np.dot(conTimes_iter,matrix_Reset)
        if (conTimes_iter - last_iter).sum() < 1:
            break
    return conTimes_iter

def getTop3(authori_list):
    v_Index = []
    for i in range(len(authori_list)):
        v_Index.append((authori_list[i],i))
    v_Index.sort(key=lambda x:x[0],reverse=True)
    #v_Index[0][1]
    return [v_Index[1][1],v_Index[2][1],v_Index[3][1],v_Index[4][1]]
```

The code below show the applying of RWR

- Experimental Results :

In our experimental the results is obtained by entering the name of the specific author that has corporation times with co-authors and after entering the specific name in the correct form the program will display the top co-authors in the clusters according to the number of the co-authors that you want to get .

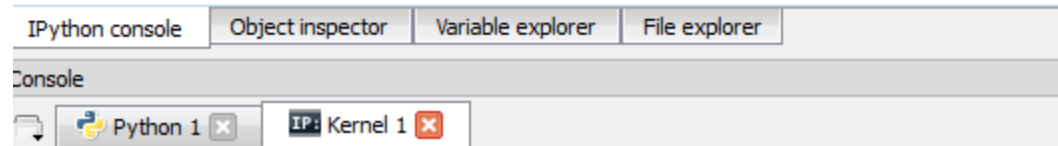
And here is the results of some queries :

```
Welcome to relational retrieve system:
=====
      input y for continue
      input n for quit

In [7]:
here: y
Prinz_WA
C:\Users\Maro_Qudar\Desktop\new_ML\class\class_91.txt
Silver_PA
Kahana_JA
Grzyb_L
Schlenstedt_G

Enter an author's name: Prinz_WA

Do you want to continue: [y]/n?: |
```



So as we can see here we started by entering “ y “ to continue then we entered a specific name for an author . Then as we can see the program has displayed the co-authors .